Comparación de dos algoritmos de ordenamiento (abbSort Vs selctionSort)

Al implementar el abbSort y el selSort encontramos grandes diferencias al momento de interpretar el código y llamar a las funciones con listas aleatorias. La primera diferencia que podemos observar es que por el modo de evaluación perezosa, el selSort podíamos pasarle listas que no superen los 4083 elementos, mientras que el abbSort no podíamos pasarle listas con mas de 1000 elementos (listas mayores a 861 elementos para ser exactos) ya que tiraba error de stack overflow. Por esto es que tuvimos que modificar el modo de evaluacion de haskell y hacer una evaluación estricta para no dejar cómputos colgados.

Además pudimos comparar los algoritmos teniendo en cuenta otros criterios que nos llevaron a determinar que el abbSort era mas eficiente que el selSort. Estos criterios que tuvimos en cuenta fueron los siguientes:

- Reducciones
- Cells
- Basura recogida
- Nro. máximo de elementos
- Tiempo

Algorithms	Time User	Reductions	Cells	Garbage	Elements
abbSort	O,145s	687371	1262893	1	1000
selSort	0,704s	6828898	9342147	9	1000
abbSort	0,219s	1386112	2543107	2	2000
selSort	2,489s	26127841	35134195	37	2000
abbSort	0,313s	2155997	3937100	4	3000
selSort	5,614s	57870966	77333911	82	3000
abbSort	0,395s	2864318	5235887	5	4000
selSort	10,384	102015787	135870081	145	4000
abbSort	0,485s	3606085	6581308	6	5000
selSort	StackOverFlow	StackOverFlow	StackOverFlow	StackOverFlow	5000
abbSort	0,973s	7545999	13690064	14	10000
selSort	StackOverFlow	StackOverFlow	StackOverFlow	StackOverFlow	10000

En conclusión al realizar las comparaciones de los resultados de ambos algoritmos de búsqueda se observo que el selSort realiza mas reducciones y comparaciones lo que lo hace mas ineficiente con respecto al tiempo gastado y desde el punto de vista teórico el selSort tiene una complejidad de N^2 y no es un algoritmo que no asegura la estabilidad de los elementos iguales.
En cambio el abbSort muestra una diferencia importante en cuanto numero de reducciones y en el tiempo utilizado por ejemplo el tiempo utilizado en el selSort para 1300 elemento es casi el mismo utilizado por el abbSort en 10000 elementos y desde el punto de vista teórico el abbSort tiene una complejidad de n.log n y es un algoritmo estable, que quiere decir que no cambia el orden relativos de elementos iguales.